WWW.SCIAM-MAGAZINE.NAROD.RU

Программирование мелодичного звучания работающего компьютера

А. К. ДЬЮДНИ

ЗВУКИ, издаваемые работающим компьютером, я впервые услышал в 1965 г. В вычислительном центре Мичиганского университета к компьютеру IBM 7090 был подключен простой электромагнитный датчик — приемник, соединенный с динамиком. По ходу выполнения компьютерной программы тот или иной регистр машины меняет свое содержимое тысячи раз в секунду. Воспринимаемое на слух щелканье регистров представляло собой удивительный поток ни на что не похожих звуков. Жужжание сменялось визгом, затем слышался треск, потом завывание. Временами звук резко переходил от басов к высоким нотам. Эти изменения звука, по-видимому, объяснялись двойным циклом в программе — внутренний цикл выполнялся все быстрее и быстрее, порождая звук все более высокого тона. В целом эта симфония напоминала дыхание кита.

С тех пор было написано много программ, позволяющих исследовать различные аспекты мелодии, гармонии и ритма. Хотя домашние компьютеры обычно не снабжаются электромагнитными датчиками, приставленными к регистру-сумматору, в большинстве случаев у них имеются небольшие динамики, подсоединенные к простейшим генераторам звуков. В наиболее распространенных языках программирования есть инструкции, выполнение которых сопровождается различными звуковыми сигналами. При помощи этого простого средства можно, с одной стороны, делать программы звучащими, а с другой — делать звучащие программы. В первом случае инструкция, генерирующая звуки, просто вставляется в программу, предназначенную для вычислений. Я остановлюсь коротко на этом случае несколько ниже. Во втором случае программа специально предназначается для создания мелодических, гармонических или ритмических эффектов. Описанные здесь две программы, пожалуй, похожи на программы первой разновидности. Лишь несколько генерирующих звук инструкций вставлены в, казалось бы, совершенно обычную последовательность инструкций.

Мелодию, рассматриваемую просто как произвольную последовательность нот, нетрудно генерировать при помощи программы. На самом деле такая программа может состоять из одного-единственного цикла. Процедура принятия решений, встроенная в цикл, определяет, какой будет следующая нота и какова будет ее продолжительность. Этот процесс можно произвольно усложнять, накладывая на него всевозможные ограничения. Мне кажется, что уже существуют программы этого типа, способные генерировать довольно складные мелодии в том или ином традиционном стиле. Я с удовольствием прочту письма читателей, которым что-либо известно о таких программах. А пока пусть известные мне алгоритмы исполнят нам свои мелодии.

Самыми скромными кандидатами на должность композитора являются простейшие арифметические алгоритмы. Выходные численные данные этих алгоритмов легко преобразуются в ноты при помощи самых разнообразных формул. Простой пример дает нам формула итерационного линейного присвоения:

$$x \leftarrow (a \cdot x + b) \bmod m.$$

Эта формула порождает целый процесс. При заданных заранее параметрах a, b и m исходное значение x преобразуется в числовую последовательность путем итерирования (повторного выполнения) формулы присвоения. Выражение «mod m» является сокращением от выражения «по модулю m», означающего, что результат вычисленного в скобках выражения должен интерпретироваться подобно показаниям стрелки часов с циферблатом из m часовых делений. Так, например, 10 по модулю 8 — это 2. Следовательно, если m равно 8 и если параметры a, b и исходное значение x — целые, то в результате итераций получаем последовательность чисел, лежащих в диапазоне от 0 до 7.

Эта числовая последовательность легко преобразуется в последовательность нот при помощи простой таблицы:

0	1	2	3	4	5	6	7
до	ре	ми	фа	соль	ля	си	до

В программе, которую я назвал SOLFEGGIO (пение по нотам), вместо названия нот элементы таблицы содержат числовые значения, соответствующие частотам (число периодов в секунду) музыкальной гаммы «до-мажор», начинающейся со среднего «до».

0	1	2	3	4	5	6	7
262	294	330	349	392	440	494	523

Всю эту программу можно представить в обычной алгоритмической записи следующим коротким алгоритмом:

```
ввести a, b, x
for i = 1 to 100
  x ← (x · a + b) mod 8
  note ← notes (x)
  сыграть note
```

В моей версии SOLFEGGIO числа в таблице *notes* (ноты) непосредственно используются инструкцией, генерирующей тон определенной высоты. В большинстве языков программирования для персональных компьютеров имеются также средства, позволяющие управлять продолжительностью звука. Пока для простоты будем считать, что эта продолжительность фиксирована и равна, скажем, половине секунды.

Теперь перед теми, кто обладает арифметическим (в отличие от музыкального) «слухом», открывается целый мир, наполненный невероятными мелодиями. В зависимости от выбора значений трех параметров формулы мы услышим либо однообразные сочинения, играющиеся как стакатто, либо странные повторяющие-

Нота	ДО	ДО♯	РЕ	РЕ♯	МИ	ФА	ФА♯	СОЛЬ	СОЛЬ♯	ЛЯ	ЛЯ♯	СИ	ДО
Частота	261,6	277,2	293,7	311,1	329,6	349,2	370,0	392,0	415,3	440,0	466,2	493,9	523,3

Частоты, соответствующие нотам выше или ниже этой октавы, получаются путем умножения или деления частот на 1,05946, т. е. на $\sqrt[12]{2}$, и округления результата

Частоты гаммы со ступенью в полтона от среднего «до» до «до» следующей октавы

WWW.SCIAM-MAGAZINE.NAROD.RU

ся мелодии на нескольких нотах, либо причудливую музыку с внезапными скачками и быстрыми фразами. К последней категории относилась и маленькая, удивительно запоминающаяся пьеса, к сожалению навсегда утерянная, потому что я не записал принятых для нее параметров. Предложенный мною выше алгоритм порождает мелодии длиной 100 нот. Возможно, будущие исследователи в этой новой музыкальной области захотят сократить эту длину, для того чтобы успеть опробовать большее число вариантов.

Конечно, программу SOLFEGGIO можно и нужно усовершенствовать. Выберем большее значение для модуля m, оно должно быть достаточно большим, чтобы охватывать две или более диатонические октавы. Это ограничение на самом деле не является жестким: вместо диатонической гаммы можно, например, воспользоваться гаммами из 12 нот, отстоящих друг от друга на полтона. Можно даже выбрать ультрасовременное музыкальное построение — «линейное присвоение», в котором числа x, полученные алгоритмом, указывают на частоты, вычисляемые путем прибавления определенной константы, скажем 100. Если в какой-то момент значение x оказывается равным 183, то программа сыграет ноту с частотой 283. Интересно, какие еще возможности придут на ум читателю? А пока вот вам занимательный вопрос. Для заданных значений a, b и m, сколько нот будет сыграно, прежде чем мелодия начнет повторяться?

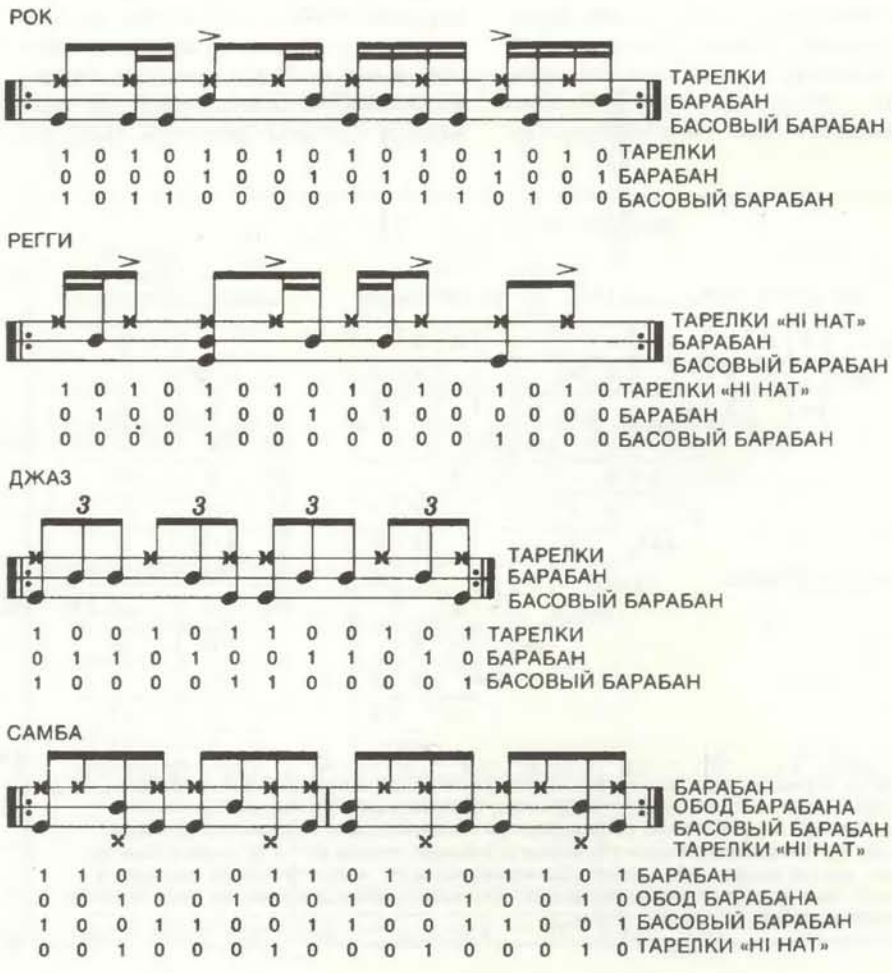
Гармония также входит теперь в сферу вокальных возможностей многих домашних компьютеров. Компьютеры, имеющие два или более динамиков, смогут исполнить весь репертуар программы, которую я назвал CANON. На самом деле энтузиасты, наверное, значительно разовьют эту идею. Существует система MIDI — цифровой интерфейс для музыкальных инструментов. Это электронный «черный ящик», преобразующий сигналы, генерируемые компьютером, в команды, управляющие электронными музыкальными инструментами, такими, как электроорган и многоканальные синтезаторы. (Статья, посвященная взаимодействию между компьютером и электронными музыкальными инструментами, опубликована в апрельском номере журнала; см. статью: М. Мэттьюз, Дж. Пирс. Компьютер в роли музыкального инструмента, «В мире науки», 1987, № 4, с. 72.)

Программа CANON в соответствии с определением музыкального термина «канон» генерирует композицию на два голоса, каждый из которых исполняет практически одну и ту же мелодию, но, как правило, на разной высоте и со сдвигом по времени. Получается очень интересная разновидность хорового пения. Канон представляет собой простейшую форму полифонии, или контрапункта (соединение в одновременное звучание двух или более мелодически самостоятельных голосов, укладывающихся в один и тот же ритм).

Такая каноническая композиция составляется из двух голосов, удовлетворяющих четырем критериям. Во-первых, все ноты имеют одну и ту же продолжительность. Во-вторых, один голос вступает вслед за другим. В-третьих, мелодии, исполняемые обоими голосами, идентичны, за исключением того, что один из голосов транспонирован вверх на некоторый стандартный музыкальный интервал (унисон, чистая кварта, чистая квинта или октава). В-четвертых, оба голоса вместе, как пары одновременно звучащих нот, должны удовлетворять определенным гармоническим правилам, изложенным в любом музыкальном учебнике. Эти правила в основном касаются дозволенных гармонических интервалов (см. рисунок на с. 100) и устанавливают форму участвующих в композиции мелодических линий. В интересах простоты последнее мы здесь опускаем.

Пример подобной композиции показан на верхнем рисунке на с. 101. Композиция была написана не машиной, а человеком. Посмотрев на нее, мы быстро замечаем, что, начиная с какого-то места, композитор уже устает от строгих ограничений, накладываемых всеми четырьмя критериями. Третье правило обычно нарушается в первую очередь — достаточным оказывается и то, что второй голос имитирует первый лишь по духу.

С другой стороны, компьютерную программу эти ограничения нисколько не угнетают. Программа CANON работает без устали, пока не завершит свою пьесу. Однако, как правило, порождаемая программой мелодия, звучит скучновато. Поэтому я сгенерировал множество коротких канонов и записал самые интересные образцы. Эти избранные фрагменты можно затем соединять в более длинные пьесы (см. нижний рисунок на с. 101). Описанный эксперимент был проведен десять лет назад с участием



Образцы ритмов для программы BEAT

WWW.SCIAM-MAGAZINE.NARODU

Г. Утаса — одного из моих студентов в Университете Западного Онтарио.

CANON сочиняет пьесу, как бы считая цифры по порядку. Пусть, например, программе было предписано сочинить крошечную композицию в стиле «канон» длиной в 6 нот. По причинам, которые вскоре станут очевидными, диапазон тонов, ограничивающих каждую мелодическую линию, невелик, скажем, 6 ступеней гаммы вверх и вниз от тоники — главного опорного звука. Множество из 13 возможных нот можно представить как цифры в системе счисления по основанию 13, или в «тринадцатиричном» счислении. Цифра 0 представляет тон, расположенный шестью ступенями ниже тоники, а «цифра» 13 — тон шестью ступенями выше тоники. Программа CANON работает очень просто, считая по порядку через все шестиразрядные числа в тринадцатеричной системе счисления. В конце концов каждое из этих чисел представляет какую-никакую, но мелодию.

Всякий раз, когда программа CANON генерирует новую мелодическую линию, она транспонирует ее вверх на квинту и переносит (по времени) на определенное количество нот, например на две. Затем она сравнивает полученные пары тонов, которые в результате предыдущей операции должны звучать одновременно. Если ни одна из полученных пар не нарушает правил гармонии, то пьеса принимается как удовлетворяющая канону и либо сразу распечатывается для сидящего у компьютера композитора, либо записывается в файл для последующей выдачи.

Программа CANON, в частности, требует, чтобы композитор перед началом выполнения программы указал следующие три параметра: *int* — интервал, на который должны отстоять друг от друга голоса, *del* — задержка, после которой вступает второй голос, и *num* — количество нот в каждой мелодической линии. Ноты генерируемой мелодии хранятся в массиве *mel*. В алгоритмической записи программу CANON можно представить в следующем виде:

```
ввести int, del, num
mel(1) ← 7
for i = 2 to num
  mel(i) ← 1
found ← «ложь»
while found = «ложь»
  увеличить mel
  for j = 1 to num − del
    сравнить mel(j) и
      mel(j + del) + int
  if они гармоничны then found ←
  ← «истина»
вывести mel
продолжить или нет
```

Алгоритм начинается с присвоения элементу *mel*(1), первой ноте канона, значения 7. Эта нота является тоникой, и она не будет меняться. Оставшиеся элементы массива *mel* все начинаются с нулевых значений. Цикл типа while проверяет значение булевской, или логической, переменной *found* (найдено), которая в исходном состоянии имеет значение «ложь». Переменной *found* присваивается значение «истина», если при выполнении остальных инструкций тела цикла найдена мелодическая линия, удовлетворяющая требованиям канона. Сначала увеличивается содержимое массива *mel*. Это делается путем просмотра массива справа налево. В процессе просмотра считающая процедура отыскивает элемент массива, значение которого меньше 13. Найдя такой элемент, она увеличивает его на 1 и устанавливает все остальные элементы справа (если они есть) в ноль. Это происходит точно так же, как и при обычном счете, только роль цифры 9 в данном случае играет «цифра» 13. Например, 3572 + 1 = 3573, 3579 + 1 = 3580 и 3599 + 1 = 3600.

Следующая операция, выполняемая в цикле, заключается в сравнении каждой ноты *mel*(*j*) с нотой *mel*(*j* + + *del*) + *int*. Другими словами, программа прибавляет значение *int* к ноте, расположенной на *del* элементов после *mel*(*j*), и по вычисленной разности между двумя тонами обращается к таблице гармонических правил, соответствующих канону. Если эта разность оказывается гармоничной во всех парах от 1 до *num* − *del*, то логической переменной *found* присваивается значение «истина». (Ноты, следующие за *num* − *del*, играются без аккомпанемента второго голоса, поэтому уже не нужно смотреть никаких разностей в таблице правил.) После того как мелодическая линия найдена и распечатана, программа спрашивает композитора: «Хотите продолжить?» Если он отвечает «да», управление в программе снова переходит на инструкцию, в которой переменной *found* присваивается значение «ложь», и программа продолжает процесс счета с того места, где она остановилась.

Можно, конечно, проигрывать мелодические сочетания, найденные программой CANON, через миниатюрный динамик собственного домашнего компьютера. Однако музыкально одаренные читатели, наверное, быстро научатся напевать сгенерированные последовательности нот или же записывать их вместе с нотами второго голоса на нотной бумаге. Тогда найденный канон можно будет опробовать на клавиатуре пианино или другого музыкального инструмента, так сказать, во всем его гармоническом великолепии.

Ритм — значительно более тонкая музыкальная форма, чем думают, наверное, многие читатели. Традицион-

a

НИЖНЯЯ НОТА \ ВЕРХНЯЯ НОТА	ДО	ДО♯	РЕ	РЕ♯	МИ	ФА	ФА♯	СОЛЬ	СОЛЬ♯	ЛЯ	ЛЯ♯	СИ	ДО
ДО		X	X			X	X				X	X	
ДО♯			X	X			X	X				X	X
РЕ				X	X			X	X				X
РЕ♯					X	X			X	X			
МИ						X	X			X	X		
ФА							X	X			X	X	
ФА♯								X	X			X	X
СОЛЬ									X	X			X
СОЛЬ♯										X	X		
ЛЯ											X	X	
ЛЯ♯												X	X
СИ													X
ДО													

b

НАЗВАНИЕ	ЧИСЛО ПОЛУ-СТУПЕНЕЙ
1	0
1+/2°	1
2	2
3−	3
3	4
Ч4	5
4+/5°	6
Ч5	7
6−	8
6	9
7−	10
7	11
8	12

Ноты, звучащие вместе, считаются консонантными согласно табл. *a*, если их комбинации не попадают в квадратики, помеченные крестиком. Названия и размеры (в полуступенях) получающихся гармонических интервалов указаны в табл. *b*. Интервалы: унисон (1), малая и большая терции (3− и 3), чистая кварта (ч4), чистая квинта (ч5), малая и большая секста (6− и 6) и октава (8) считаются консонантными по правилам канона. Соответствующие количества полутоновых (полу)ступеней: 0, 3, 4, 5, 7, 8, 9 и 12.

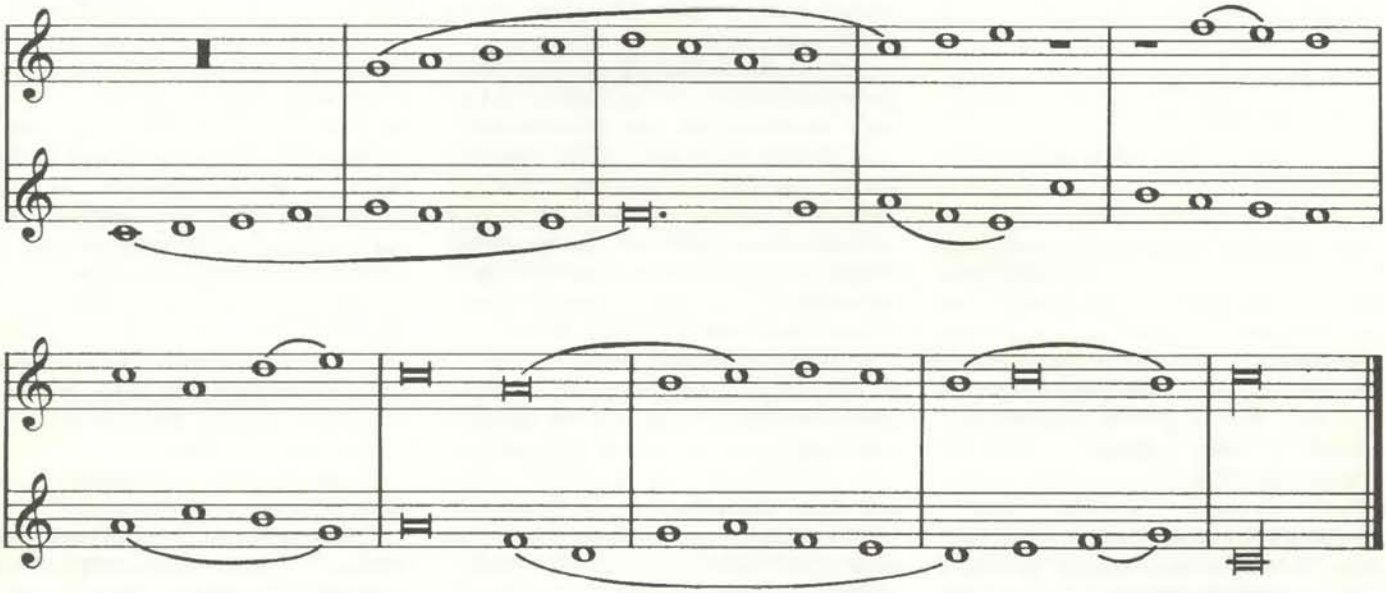
Таблица консонантных гармонических интервалов

WWW.SCIAM-MAGAZINE.NAROD.RU

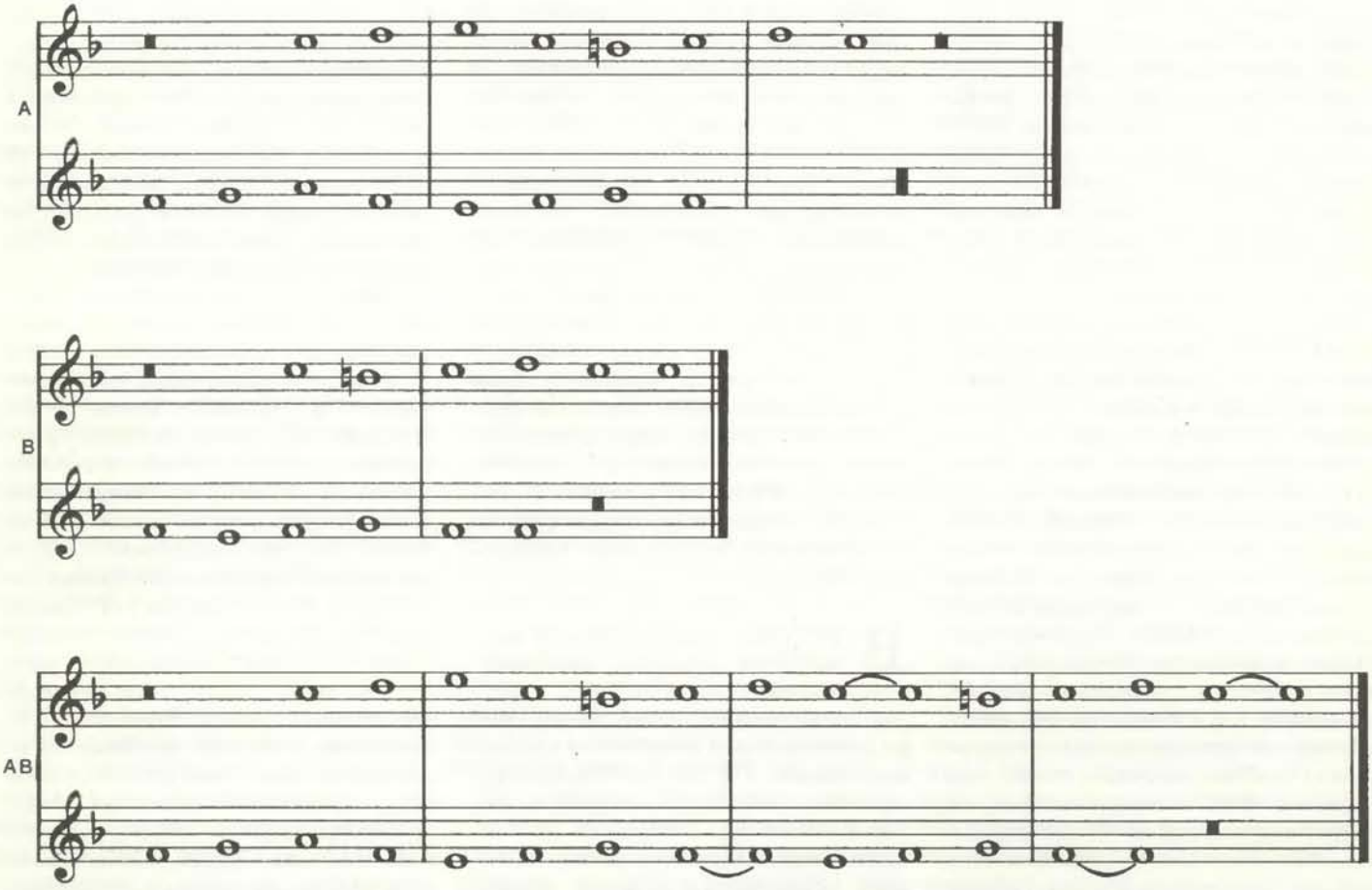
ная европейская музыка, впрочем, никогда не была ритмически изощренной. С другой стороны, современная популярная музыка вобрала в себя чрезвычайно богатое многообразие ритмических форм. Большинство этих ритмов прямо или косвенно происходят из традиционной музыки Африки или Азии. Это относится к рок-музыке, джазу, карибской и латино-американской музыке. Все более заметный вклад в современную западную музыку вносят ритмы, выбиваемые пальцами на паре барабанов. Эти ритмы происходят из индийских музыкальных форм, таких, как рага.

Программа, которую я назвал BEAT, позволяет задавать простые ритмы как последовательности нулей и единиц. Последние преобразуются в звуки по мере того, как программа повторно пробегает через последовательность. Всякий раз, встретив 1, программа посылает короткий звуковой сигнал, а 0 не сопровождается никаким сигналом.

Программа BEAT настолько проста, что ее алгоритм можно представить без всяких предисловий. Ее



Пример канона на два голоса, взятый из учебника



Два канона, сочиненные компьютером, сливаются в один

WWW.SCIAM-MAGAZINE.NAROD.RU

структура похожа на структуру программы SOLFEGGIO (программу, которая исполняет музыку в стиле «линейного присвоения»). Единственный массив *pulse* хранит ритм, заданный программистом перед началом выполнения программы.

```
ввести pulse, num, dur
for i = 1 to 25
  for j = 1 to num
  k ← 1
  while k ⩽ dur
    k ← k + 1
  if pulse(j) = 1
    then звук
```

Переменные *num* (от слова number — число) и *dur* (от слова duration — продолжительность) относятся соответственно к размеру вводимого массива и продолжительности интервала между звуковыми сигналами. Во внешнем цикле указано, что основной ритмический интервал, задаваемый массивом *pulse*, должен проигрываться 25 раз. Этот параметр можно легко изменить, если читатели встретят ритмы, которые им хотелось бы послушать подольше. Следующий внутренний цикл управляет индексом массива; алгоритм по очереди просматривает каждый элемент массива, решая посылать или не посылать звуковой сигнал. Сколько времени нужно ждать между двумя «звуковыми событиями»? Этот вопрос решается посредством специального цикла ожидания, в котором просто идет счет от 1 до значения переменной *dur*. Затем, если элемент массива *pulse*(*j*) оказывается равным 1, программа BEAT посылает звуковой сигнал; каким именно будет звук, зависит от конструктивных особенностей компьютера. Если величина *dur* мала, ритм будет быстрым. Если *dur* велико, ритм будет медленным.

Как и рассмотренные выше ее предшественницы, программа BEAT может быть усовершенствована. Можно, например, сочинять ритмы для целых ударных установок, если есть возможность ввести звуковые сигналы различного тона (низкие тона могли бы представлять большие барабаны, высокие — колокольчики и тарелки). И хотя звуки будут, по-видимому, далеки от реальных, сами по себе ритмы могут быть безупречными. Можно усложнить массив *pulse*, пользуясь в качестве его значений целыми числами 0, 1, 2 и 3, представляющими соответственно паузу, звук солирующего барабана высокого тона, звук барабана низкого тона и, наконец, одновременный удар по обоим барабанам. При этом каждое значение массива *pulse*, исследуемое во внутреннем цикле, должно быть декодировано набором инструкций «*if* ... *then*» (если ... то), которые решают, следует ли в данный момент сделать паузу, сыграть на каком-то одном «барабане» или на обоих сразу. Правда, в последнем случае два тона звучат вслед друг за другом как ближайшее возможное приближение к одновременности. Если при этом два последовательных звука будут разделены слишком большим временным интервалом, то цикл ожидания можно будет встроить внутрь конструкций «*if* ... *then*», так чтобы для каждого из четырех возможных вариантов была своя продолжительность. В случае паузы и двух последовательных звуков значение продолжительности *dur* должно быть заменено на два новых значения, одно из которых больше, а другое меньше *dur*. В обоих случаях новое значение будет отличаться от старого на величину, равную продолжительности одного звукового сигнала.

Более совершенная версия программы BEAT наверняка сможет сыграть кое-какие интересные ритмы, подобные приведенным на с. 99. Приведенные партитуры интерпретируются как последовательности дискретных событий, привязанных к одному и тому же общему отсчету времени. Вдохновленный композитор сможет легко преобразовать их в содержимое массива *pulse*.

После этих экскурсов в царство мелодии, гармонии и ритма я все же хочу вернуться к заинтриговавшей меня идее о двух ролях, которые могут играть компьютерные программы. В первой роли программа вычисляет то, что она вычисляет, и ничего более. Во второй роли программа снабжается одной-двумя звуковыми инструкциями по соседству с каждым внутренним циклом, внешним циклом и условной конструкцией (оператор if). Во втором случае программа как бы поет песню о задаче, которую решает. Тот, кто часто слушает свою излюбленную программу (будь она развлекательной или производственной), через некоторое время научится чувствовать по звуку, как она работает. У меня нет сомнения в том, что некоторые ошибки в новых версиях программы можно будет выявить на слух.

В РУБРИКЕ «Занимательный компьютер» в статье, посвященной машинкам Брайтенберга и другим электронным устройствам из мира синтетической психологии (см. «В мире науки», 1987, № 5, с. 88), я поставил перед читателями задачу попытаться объяснить поведение золотистого шара, выпавшего из неопознанного летательного объекта. Приведенный в действие шар начинает с тихим жужжанием кататься по гладкому полу. Каким образом он двигается, если у него нет никаких выступающих наружу деталей — ни колесиков, ни ножек? Шар избегает также препятствий: при встрече с препятствием он объезжает его. Каким образом он это делает?

Несколько лет назад в США и Канаде продавалась игрушка под названием «луноход». Она состояла из двух полых прозрачных пластмассовых полусфер, которые, защелкиваясь, образовывали шар. Внутрь сферы помещалась маленькая игрушечная машинка с электромоторчиком, работающим от батарейки. Чтобы привести игрушку в действие, электромотор машинки включался перед тем, как ее опускали в одну из полусфер. Шар диаметром 10 см катался по полу, потому что находящаяся внутри машинка постоянно пыталась заехать на внутреннюю стенку сферы, и последняя, выходя из статического равновесия, двигалась в том направлении, в котором в данный момент была ориентирована машинка.

Когда же сфера натыкалась на препятствие, например на ножку стула, машинка забиралась по стенке сферы еще выше. В некоторой точке начинал срабатывать эффект встроенного в правое колесо машинки проскальзывания. Машинка начинала смещаться вправо до тех пор, пока не находила свободного от препятствия направления. Таким образом сфера объезжала препятствия. Стоит лишь закрасить поверхность сферы, чтобы скрыть автомашинку, как зрители приходят в полное недоумение. Какие только фантастические предположения не высказывают люди, пытаясь угадать, что находится внутри этого таинственного шара, прибывшего к нам из космического пространства.

За время, прошедшее с момента выхода в свет нашей статьи, посвященной первому международному турниру по компьютерной игре «Бой в памяти» (см. «В мире науки», 1987, № 3, с. 96), число членов международной ассоциации любителей игры возросло, по словам ее председателя М. Кларксона, в четыре раза. Тем не менее Кларксон по-прежнему приглашает желающих присоединиться к ассоциации. В то же время У. Р. Бакли, редактор бюллетеня «Новости войны в памяти», отметил соответствующее увеличение числа подписчиков на это издание. Бакли сообщает, что бюллетень будет официальным органом ассоциации, в котором ее члены будут публиковать свои сообщения, рассказывающие о новых тактических приемах, будут высказывать свои мысли, касающиеся дальнейшего совершенствования игры и повышения класса игроков.